



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/458,121	12/08/1999	GAL MOAS	042390.P7162	8466

7590

09/20/2005

JOHN P WARD  
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP  
12400 WILSHIRE BOULEVARD 7TH FLOOR  
LOS ANGELES, CA 90025

EXAMINER
----------

VU, TUAN A

ART UNIT	PAPER NUMBER
----------	--------------

2193

DATE MAILED: 09/20/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

## Office Action Summary

Application No.

09/458,121

Applicant(s)

MOAS ET AL.

Examiner

Tuan A. Vu

Art Unit

2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

### Status

- 1) ☒ Responsive to communication(s) filed on 6/22/05.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

### Disposition of Claims

- 4) ☒ Claim(s) 1,2,4-11,13-20 and 22-25 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1,2,4-11,13-20 and 22-25 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.

### Attachment(s)

- 1) ☐ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date \_\_\_\_\_.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_.

### DETAILED ACTION

1. This action is responsive to the Applicant's response filed 6/22/2005.

As indicated in Applicant's response, claims 1, 6, 10, 15, 17, 19 and 23 have been amended. Claims 1-2, 4-11, 13-20, and 22-25 are pending in the office action.

#### *Claim Rejections - 35 USC § 103*

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1-2, 4-7, 10-11, 13-16, 19-20, and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Benson, USPN: 5,301,325 (hereinafter Benson), in view of Gosling, USPN: 5,668,999 (hereinafter Gosling).

As per claim 1, Benson discloses a method of monitoring processor resources, such method comprising: at the start of a block of code, performing just one test (e.g. col. 18, lines 12-27) for the block of code (e.g. *nodes, flow graph* - col. 12, line 21 to col. 13, line 47 - Note: flow graph visiting and nodes represented via tuple structures, each node having multiple instructions - see Fig. 6 --, such node being visited reads on at the start to blocs of code including multiple instructions), the blocs of code including multiple instructions (e.g. *flow graph ...link... tuples into node or blocks* - col. 11, lines 11-22; node 4, 5, 6 - Fig. 8; *block begin, block end* - Fig. 4; Fig. 5-6)

to determine at the start of said block of code if the resources of an architectural stack are correctly allotted for the block or have changed (e.g. *stack depth ... with the node* - col. 13, lines

Art Unit: 2193

47 to col. 14, line 5; Fig. 12; col. 18, lines 12-27 – Note: for each node, determination as to whether stack resources are appropriately allotted or have changed reads on determining if resources are correctly set to be available when needed) for the instructions to be executed; and signaling an error if said resources needed for block of code are not correctly allotted (e.g. col. 4, lines 61-66; col. 13, lines 48-57).

But Benson does not explicitly specify that the testing at the start of each block of code is for determining if the architectural stack resources needed by such block of code are available; nor does Benson expressly specify that signaling an error is in response to said stack resources (needed for the block of code) not being available. However, Benson discloses processing of block of code since starting entry point until exit points in the flow graph to determine stack depth discrepancy (e.g. col. 18, lines 13-19; Fig. 12), checking whether input/output registers are set at the beginning of head of a routine (e.g. Fig. 11 and related text); hence has taught the requirement to see to it that references needed by the block of code internal instructions for being correctly set or adjusted (e.g. col. 13, lines 28-42; Fig. 16) for execution as well as stack depth allocation ( i.e. available resources) for correct execution of instructions or allocation registers needed for a procedure. The checking to verify whether data pushed in the stack are actually available for the runtime execution of code during a pre-runtime verification is further evidenced with Gosling's disclosure. Indeed, Gosling, in a similar method to verify program code to learn about stack information and changes using analogous checking and error notifying as Benson, discloses emulating the runtime stack in order to determine whether the virtual operands, variables, or instructions, i.e. resources needed for runtime code are correctly matched with a previously recorded snapshot of the emulated stack and generate error messages when they

Art Unit: 2193

aren't matched (e.g. cols.14-16, Appendix 1; Fig 4A-G; step 440 – Fig. 4B); hence has suggested how to check if the resources that will be needed for the runtime are available and correctly so.

Based on Benson's intention to ensure that stack allocating of memory would not prevent correct execution of a block of code as mentioned above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance the graph node traversal and checking by Benson with the matching of runtime needed stack resources with pre-runtime emulated stack data, and generate error message when those resources are not available as taught by Gosling because this would enhance the method of Benson with timely integrity checking of code prior to runtime; and in so doing preclude stack overflow, and/or accommodate for data and platform discrepancies between environments in which the program is to be executed ( see Gosling: col. 1, line 33 to col. 2, line 39).

Nor does Benson explicitly disclose that that the block of code instructions have adding data to the stack or removing data from the stack. Based on the teachings to investigate stack resources at the beginning of each node (col. 18, lines 13-19; Fig. 12), and the manipulating of registers as required in a course of a tuple analysis ( see Fig. 5) as well as the *pushl*, *popl* instructions for adjusting the stack when a node is visited ( col. 12, line 65, to col. 14, line 5), the limitation as code instructions for adding to or removing data from a stack is disclosed

**As per claim 2**, Benson discloses determining a set of available resources for each block of code and check the correctness of stack allocation of such resources ( re claim 1) but does not explicitly specify determining that a set of such resources will be available after said block of code has been executed. However, Benson mentions about establishing resources state after the block of code has been executed (e.g. col. 13, lines 28-40) and what next block of code needs to

Art Unit: 2193

have checked (e.g. Fig. 16). In view of such systematic block of code traversal and in combination with the teachings by Gosling, the limitation as to determine the set of needed resources available after the previous block of code has been executed would have been obvious for the same rationale as set forth in claim 1 above; and because the checking cannot stop at one block of code to ensure the integrity of the runtime data when in opposite, all the blocks of code are to be verified.

**As per claim 4**, Benson discloses availability of stack resources as in claim 1 being determined at compile time ( *similar to a compiler* – col. 4, lines 1-38 – Note: checking stack per node visit in order to ensure whether the CC or registers are entered – see Fig. 5, 11, 12 – reads on availability of stack resources checking).

**As per claim 5**, Benson does not explicitly disclose dynamically determining the resources; but Gosling discloses the verifier to be a dynamic program to check stack proper manipulations (e.g. col. 4, lines 47-59). Official notice is taken that the use of just-in-time compiler capabilities working of bytecodes in a cross-platform runtime environment ( e.g. JVM) to expedite the execution without recommitting compilation resources was a well-known concept by the time the invention was made. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the block of code checking by Benson to include the determining of runtime needed stack resources as taught by Gosling using a dynamic verifier because this would enable Benson code to be byte code usable in byte code interpreting environments as mentioned by Gosling, thus obviating extraneous recompilation resources and extending the code usage of Benson's product as a cross platform applicability to more executing

environments as taught by well-known practice set forth above using among others JVM and just-in-time compilation/interpretation.

**As per claims 6 and 7**, Benson does not specify branching to a handler routine but Gosling discloses a exception handler routine (e.g. col. 12, lines 16-18); and it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement such handler routine to Benson's method detects a dire fault in resources allocation found in the stack especially when Benson's application program is byte codes used in a dynamic environment like that suggested by Gosling in order to address data incompatibility issue in a dynamic manner without interrupting the application process flow, thus obviating extraneous recovery network or business resources. But in case of simulation as in Gosling's teaching, simulating an exception handling routine would also have been obvious in case the whole process of verifying resources is for a pre-runtime environment.

**As per claims 10-11, and 13-16**, these claims are the computer-readable medium ( see Benson: disk 17 -Fig.2 ) or apparatus claims corresponding to method claims 1-2 and 4-7, respectively, hence are rejected herein with the same reasons as set forth above.

**As per claim 19**, Benson discloses a computer readable medium having a first set of instructions ( e.g. *front-end ...input language, input to the translator* – col. 8, lines 39-56), which when executed, generate a second set of instructions through a binary translation process, the second set of instructions (e.g. *intermediate code* – col. 11, lines 8-22; Fig. 4) when executed cause the processor to perform a method comprising the steps of:

performing only one test at the start of a block code (resources ... architectural stack are available);

the block of code including multiple instructions ( adding/removing); and  
signaling (...resources not available) just as have been recited in claim 1 above.

Hence these step limitations are rejected using the corresponding rejection of claim 1 as set forth therein, including the rationale with the obvious motivation to combine Benson and Gosling.

**As per claims 20 and 22-24**, these claims are the computer-readable medium claims corresponding to method claims 2 and 5-7, respectively, hence are rejected herein with the same reasons as set forth above.

4. Claims 8-9, 17-18, and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Benson, USPN: 5,301,325 and Gosling, USPN: 5,668,999; as applied to claims 1, 10, and 19, respectively; and further in view of Yellin et al., USPN: 5,740,441( hereinafter Yellin).

**As per claim 8**, Benson does not specify using bit vector to represent stack resources but does provide condition code and tuple to represent resources in a flow graph tree (e.g. Figs. 5-6) while Gosling discloses using exception handler but in a method to pre-verify correctness of data prior to runtime using snapshot of stack to emulate runtime data requirements similar to that of Gosling, Yellin discloses using bit vector to implement the jump subroutines with bit vector to expedite the reaching to subroutine address (e.g. col. 6, lines 6-23; Fig. 3 - Note: status array including bit reads on bit vector). To implement a dynamic checking as mentioned in claim 6 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a bit vector as taught by Yellin to the resources investigated from the stack in Benson/Gosling's combination to expedite the process of traversing the flow of block of code for verifying of routines, thus to maintain the dynamic resources integrity checking and



Art Unit: 2193

averting usage of error recovery resources during a runtime environment ( e.g. byte code in JIT machine) where possibly processor and/or volatile resources would be limited, such potential limitation being well known in remote devices ( e.g. wireless devices) in which the distributed byte codes as taught by Gosling are downloaded and executed.

**As per claim 9**, Benson and Gosling do not mention about bit vector; but Gosling mentions about simulating in a dynamic environment and using exception handling ( re claims 5 and 6). Further, in view of Yellin's teaching of bit vector in a stack emulation environment similar to Gosling, the limitation as to generate a bit vector dynamically would also have been obvious for the same reasons as mentioned in claim 8 above.

**As per claims 17-18**, these claims are the computer-readable medium claims corresponding to method claims 8-9 above, respectively, hence are rejected herein with the same reasons as set forth above.

**As per claim 25**, this claim is the computer-readable medium claim corresponding to method claim 8, and is rejected herein with the same reasons as set forth above.

### ***Response to Arguments***

5. Applicant's arguments filed 6/22/2005 have been fully considered but they are not persuasive. Following are the most representative arguments raised by Applicants and the corresponding responses by Examiner.

(A) Applicants have submitted that Benson discloses 'reporting an error message ... in response to ... different stack depths', and that Benson does not disclose '... signaling ... error if resources of an architectural stack needed ... are not available' (Appl. Rmrks, pg. 8, middle 2 paras ).

First, the rejection is based on the combined teachings of Gosling and Benson. The rejection has shown the rationale pointing to the why the teaching of Benson can be further enhanced using Gosling's checking of code elements in a stack. The fact that Gosling's stack is not architectural does not negate the usefulness as to why checking of stack allocation as by Benson can be furthered by adding to it the checking by Gosling since both methodologies are aimed at avoiding conflicts prior to executing a part of code via a flow graph analysis. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Second, the limitation recited as 'resources of the architectural stack needed for said block' is not specified in sufficient details for it to particularly enforce a teaching that clearly distinguishes from what is implied from Benson's resources checking involving a stack size or stack depths. That is, if a stack allotted depth is not correct in order for a specific code execution (like a node of code flow graph) when the code is about to be executed, the concept of resources not ready for code execution reads on availability thereof being flawed; i.e. if the initial stack is no longer the same size as the current state of the stack when such node is about to be executed, then some resource has not been allotted correctly or thus needs reallocation (or triggers alert) lest the code would run into memory fault or shortage conflict. One skill in the art would recognize a checking (stack depth checking) for such conflict eventuality as a checking so to ensure that all available stack currently allotted or previously allotted is still correct, and this is at least very suggestive of 'resources needed' for a block of code being ready, or available even

Art Unit: 2193

though Benson does not explicitly recite this using the same wording. In other words, the claims as recited and interpreted in regard to the above limitation do not preclude the scenario in which one test to check stack depth from being flagrantly dissimilar to a method for ensuring that resources are being properly available, as has been explained above.

Third, the word 'resources ... needed for a block of code' can easily be interpreted as any of the following: variable, parameters, operands as by Gosling; or as compiler dynamically gathered data like stack data, addresses, size, instrumentation data, profiling statistics etc.; and all of those are needed for enabling correct execution of code, thus when analyzed and checked prior to executing a code or entering a block of code, they read on 'resources needed'; all of which has been disclosed by corresponding parts of Benson or Gosling.

(B) Applicants have submitted that Gosling does not disclose or suggest 'signaling an error ... are not available' (Appl. Rmrks, pg. 8, middle 2 paras). The use of Gosling is to address the non-explicit teaching of 'resources ... needed for said block of code ... available'; and this has been explained in section A above. Particularly, Applicants fail to point out why it would be inappropriate to apply the code execution/memory checking based on a virtual stack ( e.g. Gosling's) to enhance an same approach using an architectural stack (e.g. Benson's ); or lay out the benefits of one stack methodology versus the other with respect to the extent of the claimed features and a broadest interpretation thereof.

(C) Applicants have submitted that claims 2, 4-9, 10, 19 are patentable over Benson in view of Gosling (Appl. Rmrks, pg. 8-9). The arguments against Benson and Gosling are addressed above; and the rejections with respect to these claims will stand.

Art Unit: 2193

(D) Applicants have submitted that claims 8-9, 17-18, and 25 are patentable over Benson in view of Gosling and Yellin (Appl. Rmrks, pg.9, middle); and that Yellin does not disclose 'signaling an error ... are not available'. The argument against Yellin does not take into consideration why Yellin has been used in the 103 type of rejection as set forth above; and appears to have attacked Yellin as if Yellin were taken alone to anticipate a limitation.

Applicants hence fail to show why the rationale as to combine Benson, Gosling and Yellin would be inappropriate. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Therefore, the rejections will stand as set forth in the Action.

#### ***Conclusion***

6. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Art Unit: 2193

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 ( for non-official correspondence – please consult Examiner before using) or 703-872-9306 ( for official correspondence) or redirected to customer service at 571-272-3609.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT  
September 08, 2005

  
KAKALI CHAKI  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2100